(4) proc perdió (in t: tablero, in j: jugadas, out res: Bool){

Pre{

juegoValido (t, j)

}

Post{

res = True ⟷ cantidadDeBombasEnPosicionesDeLaJugada (t, j) = 1

}

}

(5) proc ganó (in t: tablero, in j: jugadas, out res: Bool){

Pre{

juegoValido (t, j)

}

Post{

res = True ⟷ (cantidadDeBombasEnPosicionesDeLaJugada (t, j) = 0 ∧

jugadasTodasLasPosicionesSinBombas (t, j))

}

}

(6) proc jugar (in t: tablero, in b: banderitas, in p: pos, inout j: jugadas){

Pre{

juegoValido (t, j) ∧

posicionValidaParaNuevaJugada (|t|, p, j, b) ∧

banderitasValidaParaLaJugada (b, j, |t|) ∧

juegoEnMarcha (t, j) ∧

j = j0

}

Post{

~~j = addFirst (j0, (p, minasAdyacentes (t, p)))~~

}

podriamos definir funcion positiva...

¬ posicionNoPerteneceALaJugada (p, j) ∧

todasJugadasJ0PertenecenAJ (j0, j) ∧

|j| = |j0| + 1

CamScanner

Ejercicios

(1) aux minasAdyacentes $(t : tablero,\ p : pos) : \mathbb{Z} =$

$$\sum_{i=max(p[0]-1,\,0)}^{min(p[0]+1,\,|t|-1)} \ \sum_{j=max(p[1]-1,\,0)}^{min(p[1]+1,\,|t|-1)} es1SiPosicionEsBombaSino0(t, i, j) - es1SiPosicionEsBombaSino0(t, p[0], p[1])$$

(2) pred juegoValido (t : tablero, j : jugadas) {

(tableroValido(t) ∧

todasLasPosicionesDeLaJugadaSonValidas (|t|, j) ∧

noExistenPosicionesRepetidasEnLaJugada (j)) $\wedge_L$

(esLaCantidadDeMinasAdyacentesCorrectaParaTodaLaJugada (t, j) ∧

cantidadDeBombasEnPosicionesDeLaJugada (t, j) ≤ 1) }

(3) proc plantarBanderita (in n : ℤ, in j : jugadas, in p : pos, inout b : banderitas) {

Pre {

n ≥ 0 ∧

posicionValidaParaNuevaJugada (n, p, j, b) ∧

todasLasPosicionesDeLaJugadaSonValidas (n, j) ∧

noExistenPosicionesRepetidasEnLaJugada (j) ∧

banderitasValidaParaLaJugada (b, j, n)

$b = b_0$ }

Post {

~~b = addFirst (p, b0)~~

posicionPerteneceABanderitas (p, b) ∧

todasPosicionesDeSecuencia1PertenecenASecuencia2 (b_0, b) ∧

$|b| = |b_0| + 1$

}

}

(7) pred caminoLibre (t : tablero, p_0 : pos, p_1 : pos) {

($\exists s$: seq<pos>)((posicionPerteneceALaSecuencia (p_0, s) $\wedge$

posicionPerteneceALaSecuencia (p_1, s) $\wedge$

($\forall p$: pos)(posicionPerteneceALaSecuencia (p, s) $\rightarrow$ posiciónValida (t, p)) $\wedge_L$

($\forall p$: pos) (posicionPertenece ALaSecuencia (p, s) $\wedge$ $p \neq p_1$ $\rightarrow$ 0 = minasAdyacentes (

$\wedge$ 8 > minasAdyacentes (t, p_1) ≥ 1 $\wedge$

($\exists s_2$: seq <pos>) (secuenciaOrdenada (p_0, p_1, s_2) $\wedge$ esPermutacion(s, s_2)

}

Funciones auxiliares

pred secuenciaOrdenada (p_0, p_1 : posicion, s : seq<pos>) {

$S[0] = p_0$ $\wedge$

$S[|S|-1] = p_1$ $\wedge$

($\forall i$: $\mathbb{Z}$) ($0 \leq i < |s|-1$ $\rightarrow_L$ esAdyacente ($p[i]$, $p[i+1]$) }

- s tiene que ser una permutación de $\underbrace{\text{una secuencia ordenada.}}_{s_2}$

(8) proc jugarPlus (in t : tablero, in b : banderitas, in p : pos, inout j : jugadas) {

Pre { IGUAL AL (6) }.

Post {